

Software Project #2

ECE 1574

Fall 2005

Background:

One of the great advantages software applications provide us is the ability to simulate real-world situations that would otherwise be difficult or impossible to replicate. One such area is in the field of human behavior. It can be extremely difficult to carry out real-world experiments on human behavior because humans behave differently when they know they are being observed.

In this project you will discover how software might be used to aid in such studies by constructing a (very) simplified program for simulating the evacuation of a number of people from a room. For this first iteration you will develop a procedural application to accomplish this, which means you will only use functions and no classes.

In this iteration of this project we will increase the complexity of the rooms, allowing for walls and exits to exist in places other than the edges of the room.

Specifications:

- (1) The room will be represented by a two-dimensional matrix, one cell of which will be an exit marked by an 'E' in its spot in the input file.
- (2) The size of the room, location of the exit, the locations of any walls, and the initial locations of the people in the room will be read in from a file. Walls within the room will be marked in the input file by a 'W'.
- (3) Your program will need to maintain its own “clock.” One person will change position per clock tic. The person to move will be determined by progressing through the “room” from left to right, top to bottom. If a cell is empty, you must progress to the next one until you find one that is occupied. The clock must start at 0 and should be incremented after a person's position is updated.
- (4) A person can move to any of the 8 cells adjacent to that person's current position, or not move. A person can only be moved once per “round” (complete traversal of the room).
- (5) Two people cannot occupy the same cell.
- (6) Each person should endeavor to move to the adjacent empty cell closest to the exit. If two cells are tied for distance to the exit, then choose the first one that is found by moving counter-clockwise around the person to be moved, starting from the cell to the person's right (for example, if down-right is closest and occupied, and down and right are both tied for next closest, then choose right).
- (7) A person should not move further away from the exit than that person's current position. A person cannot move through a “wall” (either the edge of the room or a cell in the matrix marked by a 'W'). This may mean that a person is unable to move.
- (8) If the person is moved into the cell marked as the exit, that person leaves the room immediately (is removed from further consideration). This means that if two people are one square away from the exit when the first of the two is reached, both will be able to leave when they are moved.
- (9) Each time all cells have been traversed (each person considered for movement), your program must output the room, indicating which cells contain a person, and

the current clock tic.
(10) Execution ends when all people have left the room.

Grading:

80 % for program functionality

20 % for pseudocode, and program comments.

Sample Input:

5 5 //number of rows and columns, respectively

0 1 0 0 1

1 0 1 1 0

0 0 1 W 0

0 1 W 0 E

1 0 0 1 0

Sample Output:

Time: 9

0 0 1 0 0

0 1 1 0 1

0 0 1 W 1

0 0 W 1 E

0 1 0 0 0

Time: 17

0 0 0 0 0

0 0 1 1 1

0 1 1 W 0

0 0 W 0 E

0 0 1 0 0

Time: 23

0 0 0 0 0

0 0 1 1 0

0 0 0 W 1

0 1 W 1 E

0 0 0 1 0

Time: 29

0 0 0 0 0

0 0 0 0 1

0 0 1 W 0

0 0 W 0 E

0 0 1 0 0

Time: 32

0 0 0 0 0

0 0 0 0 0

0 0 0 W 1

0 0 W 1 E

0 0 0 1 0

Time: 35

0 0 0 0 0

0 0 0 0 0

0 0 0 W 0

0 0 W 0 E

0 0 0 0 0